W3School Ionic 教程

wizardforcel

Published with GitBook



目錄

W3School Ionic 教程

作者：W3School

来源：Ionic 教程

ionic 入门

ionic 简介



ionic 是一个强大的 HTML5 应用程序开发框架(HTML5 Hybrid Mobile App Framework)。 可以帮助您使用 Web 技术，比如 HTML、CSS 和 Javascript 构建接近原生体验的移动应用程序。

ionic 主要关注外观和体验，以及和你的应用程序的 UI 交互，特别适合用于基于 Hybird 模式的 HTML5 移动应用程序开发。

ionic是一个轻量的手机UI库，具有速度快，界面现代化、美观等特点。为了解决其他一些UI库在手机上运行缓慢的问题，它直接放弃了IOS6和Android4.1以下的版本支持，来获取更好的使用体验。

ionic 特点

- 1.ionic 基于Angular语法，简单易学。
- 2.ionic 是一个轻量级框架。
- 3.ionic 完美的融合下一代移动框架，支持 Angularjs 的特性， MVC ，代码易维护。
- 4.ionic 提供了漂亮的设计，通过 SASS 构建应用程序，它提供了很多 UI 组件来帮助开发者开发强大的应用。
- 5.ionic 专注原生，让你看不出混合应用和原生的区别
- 6.ionic 提供了强大的命令行工具。
- 7.ionic 性能优越，运行速度快。

学习本教程前你需要了解？

学习本教程前你需要了解以下基础知识：

- HTML
- CSS
- Javascript
- Angular

ionic 相关内容

ionic 官方网站：http://ionicframework.com/

ionic 官方文档：http://ionicframework.com/docs/

Github 地址：https://github.com/driftyco/ionic

ionic 安装

本站实例采用了ionic v1.0.1 版本，下载地址为：ionic-v1.0.1.zip。

ionic 最新版本下载地址：http://ionicframework.com/docs/overview/#download。

下载后解压压缩包，包含以下目录：

```
css/                   =>      样式文件
fonts/                 =>      字体文件
js/                    =>      Javascript文件
version.json           =>      版本更新说明
```

你也可以在 Github 上下载以下资源文件：https://github.com/driftyco/ionic（在 release 目录中）。

接下来，我们只需要在项目中引入以上目录中的 css/ionic.min.css 和 js/ionic.bundle.min.js 文件即可创建 ionic 应用。

实例

```
<ion-header-bar class="bar-positive">
    <h1 class="title">Hello World!</h1>
</ion-header-bar>

<ion-content>
    <p>我的第一个 ionic 应用。</p>
</ion-content>
```

点击 "尝试一下" 按钮查看在线实例。

本教程着重讲解 ionic 框架的应用，大部分实例在浏览器中运行，移动设备上运行可以在接下来的命令行安装教程中详细了解。

注意：在移动应用如 phonegap 中我们只需将对应的 js 和 css 文件加入到资源库中即可。

命令行安装

首先您需要安装 Node.js，我们在接下来的安装中需要使用到其 NPM 工具，更多 NPM 介绍可以查看我们的NPM 使用介绍。

然后通过命令行工具安装最新版本的 cordova 和 ionic 。通过参考Android 和 iOS 官方文档来安装。

Window 和 Linux 上打开命令行工具执行以下命令：

```
$ npm install -g cordova ionic
```

Mac 系统上使用以下命令：

```
sudo npm install -g cordova ionic
```

提示*: IOS*需要在*Mac Os X.* 和*Xcode*环境下面安装使用。

如果你已经安装了以上环境，可以执行以下命令来更新版本:

```
npm update -g cordova ionic
```

或

```
sudo npm update -g cordova ionic
```

创建应用

使用ionic官方提供的现成的应用程序模板，或一个空白的项目创建一个ionic应用：

```
$ ionic start myApp tabs
```

运行我们刚才创建的**ionic**项目

使用 ionic tool 创建，测试，运行你的apps(或者通过Cordova直接创建)。

创建android应用:

```
$ cd myApp
$ ionic platform add android
$ ionic build android
$ ionic emulate android
```

创建ios应用:

```
$ cd myApp
$ ionic platform add ios
$ ionic build ios
$ ionic emulate ios
```

ionic 创建 APP

前面的章节中我们已经学会了 ionic 框架如何导入到项目中。

接下来我们将为大家介绍如何创建一个 ionic APP 应用。

ionic 创建 APP 使用 HTML、CSS 和 Javascript 来构建，所以我们可以创建一个 www 目录，并在目录下创建 index.html 文件，代码如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Todo</title>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <link href="lib/ionic/css/ionic.css" rel="stylesheet">

    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- 在使用 Cordova/PhoneGap 创建的 APP 中包含的文件，由 Cordova/Ph
    <script src="cordova.js"></script>
  </head>
  <body>
  </body>
</html>
```

以上代码中，我们引入了 Ionic CSS 文件、 Ionic JS 文件及 Ionic AngularJS 扩展 ionic.bundle.js（ionic.bundle.js）。

ionic.bundle.js 文件已经包含了 Ionic核心JS、AngularJS、Ionic的AngularJS扩展，如果你需要引入其他 Angular 模块，可以从 lib/js/angular 目录中调用。

cordova.js 是在使用 Cordova/PhoneGap 创建应用时生成的，不需要手动引入，你可以在 Cordova/PhoneGap 项目中找到该文件，所以在开发过程中显示 404 是正常的。

创建 APP

接下来我们来实现一个包含标题、侧边栏、列表等的应用,设计图如下：



创建侧边栏

侧边栏创建使用 ion-side-menus 控制器。

编辑我们先前创建的 index.html 文件，修改 <body> 内的内容，如下所示：

```
<body>
  <ion-side-menus>
    <ion-side-menu-content>
    </ion-side-menu-content>
    <ion-side-menu side="left">
    </ion-side-menu>
  </ion-side-menus>
</body>
```

控制器解析：

- **ion-side-menus**： 是一个带有边栏菜单的容器，可以通过点击按钮或者滑动屏幕来展开菜单。

- **ion-side-menu-content**：展示主要内容的容器，可以通过滑动屏幕来展开menu。
- **ion-side-menu**：存放侧边栏的容器。

初始化 APP

接下来我们创建一个新的 AngularJS 模块，并初始化，代码位于 www/js/app.js 中：

```
angular.module('todo', ['ionic'])
```

之后在我们的 body 标签中添加 ng-app 属性：

```
<body ng-app="todo">
```

在 index.html 文件的 <script src="cordova.js"></script> 上面引入 app.js 文件：

```
<script src="js/app.js"></script>
```

修改 index.html 文件 body 标签的内容，代码如下所示：

```
<body ng-app="todo">
  <ion-side-menus>

    <!-- 中心内容 -->
    <ion-side-menu-content>
      <ion-header-bar class="bar-dark">
        <h1 class="title">Todo</h1>
      </ion-header-bar>
      <ion-content>
      </ion-content>
    </ion-side-menu-content>

    <!-- 左侧菜单 -->
    <ion-side-menu side="left">
      <ion-header-bar class="bar-dark">
        <h1 class="title">Projects</h1>
      </ion-header-bar>
    </ion-side-menu>

  </ion-side-menus>
</body>
```

以上步骤我们已经完成了 ionic 基本 APP 的应用。

创建列表

首先创建一个控制器 **TodoCtrl**：

```
<body ng-app="todo" ng-controller="TodoCtrl">
```

在app.js文件中，使用控制器定义列表数据：

```
angular.module('todo', ['ionic'])

.controller('TodoCtrl', function($scope) {
  $scope.tasks = [
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' },
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' }
  ];
});
```

在index.html页面中数据列表我们使用 Angular ng-repeat 来迭代数据：

```
<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <h1 class="title">Todo</h1>
  </ion-header-bar>
  <ion-content>
    <!-- 列表 -->
    <ion-list>
      <ion-item ng-repeat="task in tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>
```

修改后 index.html body 标签内代码如下：

```
 <body ng-app="todo" ng-controller="TodoCtrl">
    <ion-side-menus>

    <!-- 中心内容 -->
    <ion-side-menu-content>
      <ion-header-bar class="bar-dark">
        <h1 class="title">Todo</h1>
      </ion-header-bar>
      <ion-content>
        <!-- 列表 -->
        <ion-list>
          <ion-item ng-repeat="task in tasks">
            {{task.title}}
          </ion-item>
        </ion-list>
      </ion-content>
    </ion-side-menu-content>

    <!-- 左侧菜单 -->
    <ion-side-menu side="left">
      <ion-header-bar class="bar-dark">
        <h1 class="title">Projects</h1>
      </ion-header-bar>
    </ion-side-menu>

    </ion-side-menus>
    <script src="http://www.runoob.com/static/ionic/js/app.js"></sc
    <script src="cordova.js"></script>
</body>
```

创建添加页面

修改 index.html 在 **</ion-side-menus>** 后添加如下代码:

```
<script id="new-task.html" type="text/ng-template">

  <div class="modal">

    <!-- Modal header bar -->
    <ion-header-bar class="bar-secondary">
      <h1 class="title">New Task</h1>
      <button class="button button-clear button-positive" ng-click=
    </ion-header-bar>

    <!-- Modal content area -->
    <ion-content>

      <form ng-submit="createTask(task)">
        <div class="list">
          <label class="item item-input">
            <input type="text" placeholder="What do you need to do?
          </label>
        </div>
        <div class="padding">
          <button type="submit" class="button button-block button-p
        </div>
      </form>

    </ion-content>

  </div>

</script>
```

以上代码中我们通过 **<script id="new-task.html" type="text/ng-template">** 定义了新的模板页面。

表单提交时触发 createTask(task) 函数。

ng-model="task.title" 会将表单的输入数据赋值给 task 对象的 title 属性。

修改 **<ion-side-menu-content>** 内的内容，代码如下：

```
<!-- 中心内容 -->
<ion-side-menu-content>
<ion-header-bar class="bar-dark">
  <h1 class="title">Todo</h1>
  <!-- 新增按钮-->
  <button class="button button-icon" ng-click="newTask()">
    <i class="icon ion-compose"></i>
  </button>
</ion-header-bar>
<ion-content>
  <!-- 列表 -->
  <ion-list>
    <ion-item ng-repeat="task in tasks">
      {{task.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu-content>
```

app.js 文件中，控制器代码如下：

```
angular.module('todo', ['ionic'])

.controller('TodoCtrl', function($scope, $ionicModal) {
  $scope.tasks = [
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' },
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' }
  ];

  // 创建并载入模型
  $ionicModal.fromTemplateUrl('new-task.html', function(modal) {
    $scope.taskModal = modal;
  }, {
    scope: $scope,
    animation: 'slide-in-up'
  });

  // 表单提交时调用
  $scope.createTask = function(task) {
    $scope.tasks.push({
      title: task.title
    });
    $scope.taskModal.hide();
    task.title = "";
  };

  // 打开新增的模型
  $scope.newTask = function() {
    $scope.taskModal.show();
  };

  // 关闭新增的模型
  $scope.closeNewTask = function() {
    $scope.taskModal.hide();
  };
});
```

创建侧边栏

通过以上步骤我们已经实现了新增功能，现在我们为 app 添加侧边栏功能。

修改 内的内容，代码如下：

```
<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <button class="button button-icon" ng-click="toggleProjects()">
      <i class="icon ion-navicon"></i>
    </button>
    <h1 class="title">{{activeProject.title}}</h1>
    <!-- 新增按钮 -->
    <button class="button button-icon" ng-click="newTask()">
      <i class="icon ion-compose"></i>
    </button>
  </ion-header-bar>
  <ion-content scroll="false">
    <ion-list>
      <ion-item ng-repeat="task in activeProject.tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>
```

添加侧边栏：

```
<!-- 左边栏 -->
<ion-side-menu side="left">
<ion-header-bar class="bar-dark">
  <h1 class="title">Projects</h1>
  <button class="button button-icon ion-plus" ng-click="newProject(
  </button>
</ion-header-bar>
<ion-content scroll="false">
  <ion-list>
    <ion-item ng-repeat="project in projects" ng-click="selectProje
      {{project.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu>
```

<ion-item> 中的 ng-class 指令设置菜单激活样式。

这里我们创建一个新的js 文件 app2.js(为了不与前面的混淆)，代码如下：

```
angular.module('todo', ['ionic'])
/**
 * The Projects factory handles saving and loading projects
 * from local storage, and also lets us save and load the
```

```
 * last active project index.
 */
.factory('Projects', function() {
  return {
    all: function() {
      var projectString = window.localStorage['projects'];
      if(projectString) {
        return angular.fromJson(projectString);
      }
      return [];
    },
    save: function(projects) {
      window.localStorage['projects'] = angular.toJson(projects);
    },
    newProject: function(projectTitle) {
      // Add a new project
      return {
        title: projectTitle,
        tasks: []
      };
    },
    getLastActiveIndex: function() {
      return parseInt(window.localStorage['lastActiveProject']) ||
    },
    setLastActiveIndex: function(index) {
      window.localStorage['lastActiveProject'] = index;
    }
  }
})

.controller('TodoCtrl', function($scope, $timeout, $ionicModal, Pro

  // A utility function for creating a new project
  // with the given projectTitle
  var createProject = function(projectTitle) {
    var newProject = Projects.newProject(projectTitle);
    $scope.projects.push(newProject);
    Projects.save($scope.projects);
    $scope.selectProject(newProject, $scope.projects.length-1);
  }

  // Load or initialize projects
  $scope.projects = Projects.all();

  // Grab the last active, or the first project
  $scope.activeProject = $scope.projects[Projects.getLastActiveInde

  // Called to create a new project
  $scope.newProject = function() {
    var projectTitle = prompt('Project name');
    if(projectTitle) {
      createProject(projectTitle);
    }
```

```
  };

  // Called to select the given project
  $scope.selectProject = function(project, index) {
    $scope.activeProject = project;
    Projects.setLastActiveIndex(index);
    $ionicSideMenuDelegate.toggleLeft(false);
  };

  // Create our modal
  $ionicModal.fromTemplateUrl('new-task.html', function(modal) {
    $scope.taskModal = modal;
  }, {
    scope: $scope
  });

  $scope.createTask = function(task) {
    if(!$scope.activeProject || !task) {
      return;
    }
    $scope.activeProject.tasks.push({
      title: task.title
    });
    $scope.taskModal.hide();

    // Inefficient, but save all the projects
    Projects.save($scope.projects);

    task.title = "";
  };

  $scope.newTask = function() {
    $scope.taskModal.show();
  };

  $scope.closeNewTask = function() {
    $scope.taskModal.hide();
  }

  $scope.toggleProjects = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };

  // Try to create the first project, make sure to defer
  // this by using $timeout so everything is initialized
  // properly
  $timeout(function() {
    if($scope.projects.length == 0) {
      while(true) {
        var projectTitle = prompt('Your first project title:');
        if(projectTitle) {
          createProject(projectTitle);
          break;
```

```
        }
      }
    }
  });

});
```

body 中 ion-side-menus 代码如下：：

```
<ion-side-menus>

<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <button class="button button-icon" ng-click="toggleProjects()">
      <i class="icon ion-navicon"></i>
    </button>
    <h1 class="title">{{activeProject.title}}</h1>
    <!-- 新增按钮 -->
    <button class="button button-icon" ng-click="newTask()">
      <i class="icon ion-compose"></i>
    </button>
  </ion-header-bar>
  <ion-content scroll="false">
    <ion-list>
      <ion-item ng-repeat="task in activeProject.tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>

<!-- 左边栏 -->
<ion-side-menu side="left">
<ion-header-bar class="bar-dark">
  <h1 class="title">Projects</h1>
  <button class="button button-icon ion-plus" ng-click="newProject(
  </button>
</ion-header-bar>
<ion-content scroll="false">
  <ion-list>
    <ion-item ng-repeat="project in projects" ng-click="selectProje
      {{project.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu>

</ion-side-menus>
```

ionic CSS

ionic 头部与底部

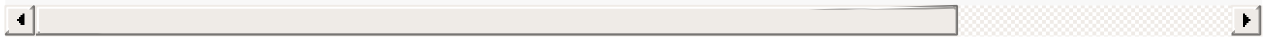
Header(头部)

Header是固定在屏幕顶部的组件,可以包如标题和左右的功能按钮。

ionic 默认提供了许多种颜色样式，你可以调用不同的样式名，当然也可以自定义一个。

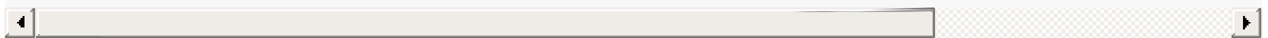
bar-light

```
<div  class="bar bar-header bar-light">  <h1  class="title">bar-lig
```



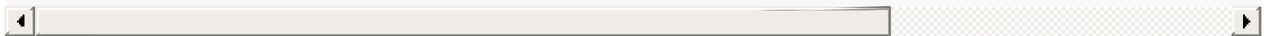
bar-stable

```
<div  class="bar bar-header bar-stable">  <h1  class="title">bar-st
```



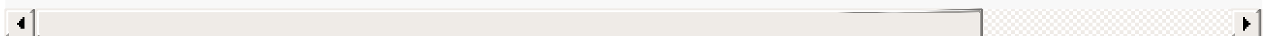
bar-positive

```
<div  class="bar bar-header bar-positive">  <h1  class="title">bar-
```



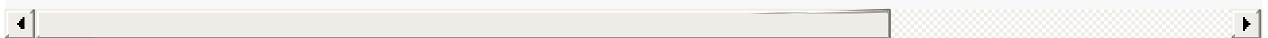
bar-calm

```
<div  class="bar bar-header bar-calm">  <h1  class="title">bar-calm
```



bar-balanced

```
<div  class="bar bar-header bar-balanced">  <h1  class="title">bar-
```



bar-energized

```
<div  class="bar bar-header bar-energized">  <h1  class="title">bar
```

bar-assertive

```
<div  class="bar bar-header bar-assertive">  <h1  class="title">bar
```

bar-royal

```
<div  class="bar bar-header bar-royal">  <h1  class="title">bar-roy
```

bar-dark

```
<div  class="bar bar-header bar-dark">  <h1  class="title">bar-dark
```

Sub Header（副标题）

Sub Header同样是固定在顶部，只是是在Header的下面，就算没有写Header这个，Sub Header这个样式也会距离顶部有一个Header的距离。颜色样式同 Header 。

```
<div  class="bar bar-header">  <h1  class="title">Header</h1>  </di
```

Footer(底部)

Footer 是在屏幕的最下方，可以包含多种内容类型。

```
<div  class="bar bar-footer bar-balanced">  <div  class="title">Foo
```

Footer 同上面的 Header，只是把样式名 bar-header 换做 bar-footer 。

```
<div  class="bar bar-footer">  <button  class="button button-clear'
```

此外，如果底部没有标题，但是又需要右边的按钮，你需要在右侧按钮添加 pull-right如:

```
<div  class="bar bar-footer">  <button  class="button button-clear
```

ionic 按钮

按钮是移动app不可或缺的一部分，不同风格的app，需要的不同按钮的样式。

默认情况下，按钮显示样式为：**display: inline-block**。

```
<button class="button">
  Default
</button>

<button class="button button-light">
  button-light
</button>

<button class="button button-stable">
  button-stable
</button>

<button class="button button-positive">
  button-positive
</button>

<button class="button button-calm">
  button-calm
</button>

<button class="button button-balanced">
  button-balanced
</button>

<button class="button button-energized">
  button-energized
</button>

<button class="button button-assertive">
  button-assertive
</button>

<button class="button button-royal">
  button-royal
</button>

<button class="button button-dark">
  button-dark
</button>
```

button-block 样式按钮显示为：**display: block**，它将完全填充父元素的宽度，包含了内边距属性padding。

```
<button class="button button-block button-positive">
  Block Button
</button>
```

使用 button-full 类，可以让按钮显示完全宽度，且不包含内边距padding。

```
<button class="button button-full button-positive">
  Full Width Block Button
</button>
```

不同大小的按钮

button-large 设置为大按钮，button-small 设置为小按钮。

```
<button class="button button-small button-assertive">
  Small Button
</button>
<button class="button button-large button-positive">
  Large Button
</button>
```

无背景按钮

button-outline 设置背景为透明。

```
<button class="button button-outline button-positive">
  Outlined Button
</button>
```

无背景与边框按钮

button-clear 设置按钮背景为透明，且无边框。

```
<button class="button button-clear button-positive">
  Clear Button
</button>
```

图标按钮

我们可以在按钮上添加图标。

```
<button class="button">
  <i class="icon ion-loading-c"></i> Loading...
</button>

<button class="button icon-left ion-home">Home</button>

<button class="button icon-left ion-star button-positive">Favorites

<a class="button icon-right ion-chevron-right button-calm">Learn Mo

<a class="button icon-left ion-chevron-left button-clear button-dar

<button class="button icon ion-gear-a"></button>

<a class="button button-icon icon ion-settings"></a>

<a class="button button-outline icon-right ion-navicon button-balar
```

头部/底部添加按钮

头部/底部可以添加按钮，按钮的样式根据头部/底部来设定，所以你不需要为按钮添加额外的样式。

```
<div class="bar bar-header">
  <button class="button icon ion-navicon"></button>
  <h1 class="title">Header Buttons</h1>
  <button class="button">Edit</button>
</div>
```

button-clear 类可以设置无背景和边框的头部/底部按钮。

```
<div class="bar bar-header">
  <button class="button button-icon icon ion-navicon"></button>
  <div class="h1 title">Header Buttons</div>
  <button class="button button-clear button-positive">Edit</button>
</div>
```

按钮栏

我们可以使用 button-bar 类来设置按钮栏。以下实例中，我们在头部和内容中添加了按钮栏。

```
<div class="button-bar">
  <a class="button">First</a>
  <a class="button">Second</a>
  <a class="button">Third</a>
</div>
```

ionic 列表

列表是一个应用广泛的界面元素，在所有移动app中几乎都会使用到。

列表可以是基本文字、按钮，开关，图标和缩略图等。

列表项可以是任何的HTML元素。容器元素需要list类，每个列表项需要使用item类。

ionList和ionItem可以很容易的支持各种交互方式，比如，滑动编辑，拖动排序，以及删除项。

基本用法:

```
<ul class="list">
    <li class="item">
      ...
    </li>
</ul>
```

列表分隔符

我们可以使用 item-divider 类来为列表创建分隔符，默认情况下，列表项以不同的背景颜色和字体加粗来区分，但你也可以很容易的定制他。

```
<div class="list">

  <div class="item item-divider">
    Candy Bars
  </div>

  <a class="item" href="#">
    Butterfinger
  </a>

  ...

</div>
```

带图标列表

我们可以在列表项的左侧或右侧指定图标。

使用 item-icon-left 图标在左侧， item-icon-right 设置图标在右侧。如果你需要在两边都有图标，则两个类都添加上即可。

以下实例中，我们在列表项中使用了 <a> 标签，使得每个列表项可点击。

列表项在使用 <a>或<button> 元素时，如果右侧未添加图标，则会自动添加上箭头号。

实例中，第一项只有左侧图标，第二项左右均有图标，第三项有右侧图标（还有注释 item-note），第四项有badge（标记）元素。

```
<div class="list">

  <a class="item item-icon-left" href="#">
    <i class="icon ion-email"></i>
    Check mail
  </a>

  <a class="item item-icon-left item-icon-right" href="#">
    <i class="icon ion-chatbubble-working"></i>
    Call Ma
    <i class="icon ion-ios-telephone-outline"></i>
  </a>

  <a class="item item-icon-left" href="#">
    <i class="icon ion-mic-a"></i>
    Record album
    <span class="item-note">
      Grammy
    </span>
  </a>

  <a class="item item-icon-left" href="#">
    <i class="icon ion-person-stalker"></i>
    Friends
    <span class="badge badge-assertive">0</span>
  </a>

</div>
```

按钮列表

使用 item-button-right 或 item-button-left 类将按钮放在列表项中。

```
<div class="list">

  <div class="item item-button-right">
    Call Ma
    <button class="button button-positive">
      <i class="icon ion-ios-telephone"></i>
    </button>
  </div>

  ...

</div>
```

带头像列表

使用 item-avatar 来创建一个带头像的列表：

```
<div class="list">

    <a class="item item-avatar" href="#">
      <img src="venkman.jpg">
      <h2>Venkman</h2>
      <p>Back off, man. I'm a scientist.</p>
    </a>

    ...

</div>
```

缩略图列表

item-thumbnail-left 类用于添加左侧对齐的缩略图， item-thumbnail-right 类用于添加右侧对齐的缩略图。

```
<div class="list">

    <a class="item item-thumbnail-left" href="#">
      <img src="cover.jpg">
      <h2>Pretty Hate Machine</h2>
      <p>Nine Inch Nails</p>
    </a>

    ...

</div>
```

内嵌列表(inset list)

我们可以在容器当中内嵌列表，列表不会显示完整的宽度。

内嵌列表的样式为：list list-inset，与常规列表区别是，它设置了外边距（marign）,类似于选项卡。

内嵌列表是没有阴影效果的，滚动时效果会更好。

```
<div class="list list-inset">

    <div class="item">
      Raiders of the Lost Ark
    </div>

    ...

</div>
```

ionic 卡片

近年来卡片(card)的应用越来越流行，卡片提供了一个更好组织信息展示的工具。

针对移动端的应用，卡片会根据屏幕大小自适应大小。

我们可以很灵活的控制卡片的显示效果，甚至实现动画效果。

卡片一般放在页面顶部，当然也可以实现左右切换的功能。

```
<div class="card">
  <div class="item item-text-wrap">
    基本卡片，包含了文本信息。
  </div>
</div>
```

卡片(card)默认样式带有box-shadow(阴影)，由于性能的原因，和他类似的元素像 list list-inset 并没有阴影。

如果你有很多的卡片，每个卡片都有很多子元素，建议使用内嵌列表（inset list）。

卡片的头部与底部

我们可以通过添加 item-divider 类为卡片添加头部与底部：

```
<div class="card">
  <div class="item item-divider">
    卡片头部！
  </div>
  <div class="item item-text-wrap">
    基本卡片，包含了文本信息。
  </div>
  <div class="item item-divider">
    卡片底部！
  </div>
</div>
```

卡片列表

使用 list card 类来设置卡片列表：

```
<div class="list card">

  <a href="#" class="item item-icon-left">
    <i class="icon ion-home"></i>
    Enter home address
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-ios-telephone"></i>
    Enter phone number
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-wifi"></i>
    Enter wireless password
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-card"></i>
    Enter card information
  </a>

</div>
```

带图片卡片

卡片中使用图片，效果会更好，实例如下：

```
<div class="list card">

  <div class="item item-avatar">
    <img src="avatar.jpg">
    <h2>Pretty Hate Machine</h2>
    <p>Nine Inch Nails</p>
  </div>

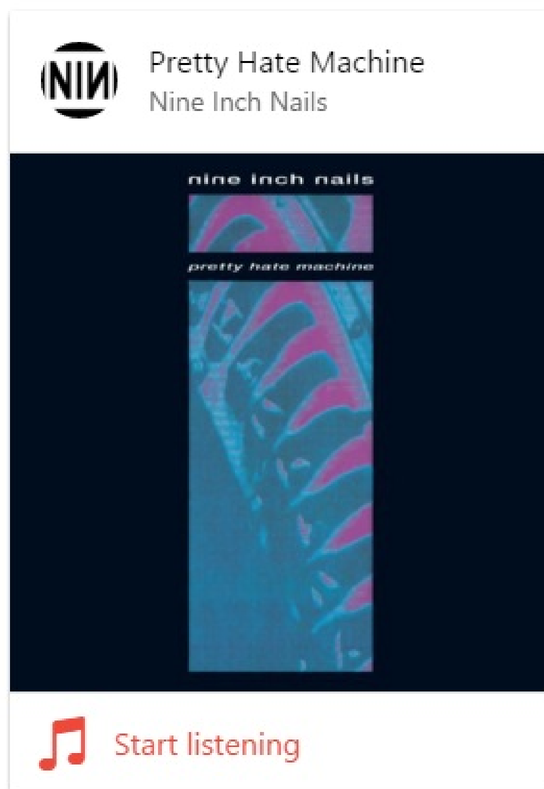
  <div class="item item-image">
    <img src="cover.jpg">
  </div>

  <a class="item item-icon-left assertive" href="#">
    <i class="icon ion-music-note"></i>
    Start listening
  </a>

</div>
```

运行效果如下：

卡片



卡片展现

以下实例中使用几种不同的选项的卡片展现方式。 开始使用了 list card 元素，并使用了 item-avatar , item-body 元素用于展示图片和文本信息，底部使用 item-divider 类。

```
<div class="list card">

  <div class="item item-avatar">
    <img src="mcfly.jpg">
    <h2>Marty McFly</h2>
    <p>November 05, 1955</p>
  </div>

  <div class="item item-body">
    <img class="full-image" src="delorean.jpg">
    <p>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！
    </p>
    <p>
      <a href="#" class="subdued">1 喜欢</a>
      <a href="#" class="subdued">5 评论</a>
    </p>
  </div>

  <div class="item tabs tabs-secondary tabs-icon-left">
    <a class="tab-item" href="#">
      <i class="icon ion-thumbsup"></i>
      喜欢
    </a>
    <a class="tab-item" href="#">
      <i class="icon ion-chatbox"></i>
      Comment
    </a>
    <a class="tab-item" href="#">
      <i class="icon ion-share"></i>
      分享
    </a>
  </div>

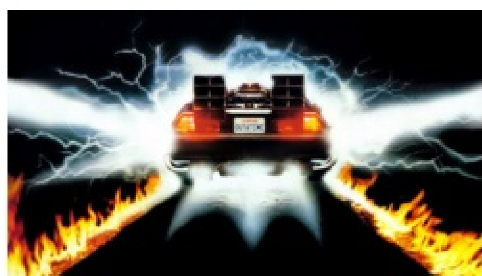
</div>
```

运行效果如下：

卡片



Marty McFly
November 05, 1955



菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！

1 喜欢 5 评论

 喜欢  评论  分享

ionic 表单和输入框

list 类同样可以用于 input 元素。item-input 和 item 类指定了文本框及其标签。

输入框属性：placeholder

以下实例中，默认为100%宽度（左右两侧没有边框），并使用 placeholder 属性设置输入字段预期值的提示信息。

```
<div class="list">
  <label class="item item-input">
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input">
    <input type="text" placeholder="Last Name">
  </label>
  <label class="item item-input">
    <textarea placeholder="Comments"></textarea>
  </label>
</div>
```

输入框属性：input-label

使用 input-label 将标签放置于输入框 input 的左侧。

```
<div class="list">
  <label class="item item-input">
    <span class="input-label">用户名：</span>
    <input type="text">
  </label>
  <label class="item item-input">
    <span class="input-label">密码：</span>
    <input type="password">
  </label>
</div>
```

堆叠标签

堆叠标签通常位于输入框的头部。每个选项使用 item-stacked-label 类指定。 每个输入框需要指定 input-label。以下实例也使用了 placeholder 属性来设置信息输入提示。

```
<div class="list">
  <label class="item item-input item-stacked-label">
    <span class="input-label">First Name</span>
    <input type="text" placeholder="John">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Last Name</span>
    <input type="text" placeholder="Suhr">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Email</span>
    <input type="text" placeholder="john@suhr.com">
  </label>
</div>
```

浮动标签

浮动标签类似于堆叠标签，但浮动标签有一个动画的效果，每个选项需要指定 item-floating-label 类，输入标签需要指定 input-label。

```
<div class="list">
  <label class="item item-input item-floating-label">
    <span class="input-label">First Name</span>
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input item-floating-label">
    <span class="input-label">Last Name</span>
    <input type="text" placeholder="Last Name">
  </label>
  <label class="item item-input item-floating-label">
    <span class="input-label">Email</span>
    <input type="text" placeholder="Email">
  </label>
</div>
```

内嵌表单

默认情况下每个输入域宽度都是100%，但我们可以使用 list list-inset 或 card 类设置表单的内边距(padding)， card 类带有阴影。

```
<div class="list list-inset">
  <label class="item item-input">
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input">
    <input type="text" placeholder="Last Name">
  </label>
</div>
```

内嵌输入域

使用 list-inset 设置内嵌实体列表。 使用 item-input-inset 样式可以内嵌一个按钮。

```
<div class="list">

  <div class="item item-input-inset">
    <label class="item-input-wrapper">
      <input type="text" placeholder="Email">
    </label>
    <button class="button button-small">
      Submit
    </button>
  </div>

</div>
```

带图标的输入框

item-input 输入框可以很简单的添加图标。 图标可以在 <input> 前添加。

```
<div class="list list-inset">
  <label class="item item-input">
    <i class="icon ion-search placeholder-icon"></i>
    <input type="text" placeholder="Search">
  </label>
</div>
```

头部输入框

输入框可放置在头部，并可添加提交或取消按钮。

```
<div class="bar bar-header item-input-inset">
  <label class="item-input-wrapper">
    <i class="icon ion-ios-search placeholder-icon"></i>
    <input type="search" placeholder="搜索">
  </label>
  <button class="button button-clear">
    取消
  </button>
</div>
```

ionic Toggle(切换开关)

切换开关类似与 HTML 的 checkbox 标签，但它更易于在移动设备上使用。

切换开关可以使用 toggle-assertive 来指定颜色。

```
<label class="toggle">
   <input type="checkbox">
   <div class="track">
     <div class="handle"></div>
   </div>
</label>
```

该实例有是多个切换开关列表。注意，每个选项的 item 类后需要添加 item-toggle 类。

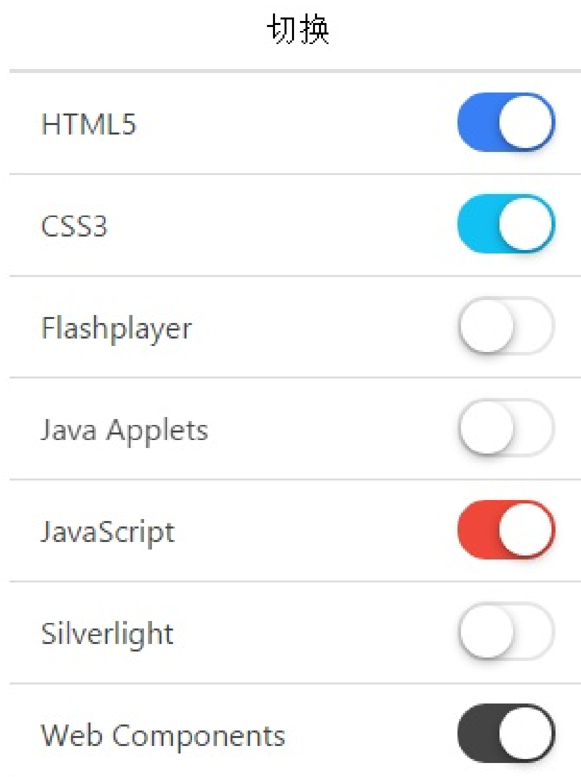
```
<ul class="list">

  <li class="item item-toggle">
     HTML5
     <label class="toggle toggle-assertive">
       <input type="checkbox">
       <div class="track">
         <div class="handle"></div>
       </div>
     </label>
  </li>

  ...

</ul>
```

运行效果如下：



ionic checkbox（复选框）

ionic 里面的 Checkbox 和普通的 Checkbox 效果上其实相差不大，只是样式上有所不同。

以下实例颜色了多个复选框的列表。

注意，每个选项的 item 类后需要添加 item-checkbox 类。

复选框可以使用 checkbox-assertive 来指定颜色。

```
<ul class="list">

  <li class="item item-checkbox">
     <label class="checkbox">
       <input type="checkbox">
     </label>
     Flux Capacitor
  </li>

  ...

</ul>
```

运行效果如下：

复选框

 Flux Capacitor

 1.21 Gigawatts

 Delorean

 88 MPH

 Plutonium Resupply

ionic 单选框

ionic 当选按钮与标准 type="radio" 的 input元素类似。以下展示了现代app类型的单选按钮。

每个 item-radio 中的 type="radio" 的 input 元素的 name 属性都相同。radio-icon 类用于是否显示图标。

ionic 在单选项中使用了 <label> 元素，使其更易点击。

实例

```
<div class="list">

<label class="item item-radio">
  <input type="radio" name="group" value="go" checked="checked">
  <div class="item-content">
    Go
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="python">
  <div class="item-content">
    Python
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="ruby">
  <div class="item-content">
    Ruby
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value=".net">
  <div class="item-content">
    .Net
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="java">
```

```
  <div class="item-content">
    Java
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="php">
  <div class="item-content">
    PHP
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

</div>
```

运行效果如下：

单选按钮列表

Go ✔

Python

Ruby

.Net

Java

PHP

ionic Range

ionic Range 是一个滑块控件，ionic 为 Range 提供了很多种默认的样式。而且你可以在许多种元素里使用它比如列表或者 Card 。

实例

```
<div class="range">
  <i class="icon ion-volume-low"></i>
  <input type="range" name="volume">
  <i class="icon ion-volume-high"></i>
</div>

<div class="list" style="margin-top: 13px">
  <div class="item item-divider">
    Ranges In A List
  </div>
  <div class="item range range-positive">
    <i class="icon ion-ios-sunny-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="12">
    <i class="icon ion-ios-sunny"></i>
  </div>
  <div class="item range range-calm">
    <i class="icon ion-ios-lightbulb-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="25">
    <i class="icon ion-ios-lightbulb"></i>
  </div>
  <div class="item range range-balanced">
    <i class="icon ion-ios-bolt-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="38">
    <i class="icon ion-ios-bolt"></i>
  </div>
  <div class="item range range-energized">
    <i class="icon ion-ios-moon-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="50">
    <i class="icon ion-ios-moon"></i>
  </div>
  <div class="item range range-assertive">
    <i class="icon ion-ios-partlysunny-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="63">
    <i class="icon ion-ios-partlysunny"></i>
  </div>
  <div class="item range range-royal">
    <i class="icon ion-ios-rainy-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="75">
    <i class="icon ion-ios-rainy"></i>
  </div>
  <div class="item range range-dark">
    <i class="icon ion-ios-lightbulb-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="88">
    <i class="icon ion-ios-lightbulb"></i>
  </div>
</div>
```

运行效果如下：

Range（滑块控件）

Ranges In A List

ionic select

ionic select 的 select 相比原生的要更加美观些。但是弹出的可选选项样式是浏览器默认的。

每个平台上的可选项样式都是不一样的，在PC电脑的浏览器上，你会看到传统的下拉界面，Android 上会弹出单选按钮选项，iOS 有个滚轮操作界面。

实例

```
<div class="list"> <div class="item item-input item-select"> <
```

运行效果如下：

Select

Lightsaber	Green
Fighter	X-wing
Droid	R2-D2
Planet	Dagobah

ionic tab(选项卡)

ionic tab(选项卡) 是水平排列的按钮或者链接，用以页面间导航的切换。它可以包含文字和图标的组合，是一种移动设备上流行的导航方法。

以下选项卡容器使用了 tabs 类，每个选项卡使用 tab-item 类。默认情况下，选项卡是文本的，并没有图标。

实例

```
<div class="tabs">
  <a class="tab-item">
    主页
  </a>
  <a class="tab-item">
    收藏
  </a>
  <a class="tab-item">
    设置
  </a>
</div>
```

默认情况，选项卡颜色为默认，你可以设置以下不同颜色样式：tabs-default，tabs-light， tabs-stable， tabs-positive， tabs-calm， tabs-balanced， tabs-energized， tabs-assertive， tabs-royal， tabs-dark。

要隐藏选项卡栏，可使用 tabs-item-hide 类。

图标选项卡

在 tabs 类后添加 tabs-icon-only 类可设置只显示图标选项卡。

```
<div class="tabs tabs-icon-only">
  <a class="tab-item">
    <i class="icon ion-home"></i>
  </a>
  <a class="tab-item">
    <i class="icon ion-star"></i>
  </a>
  <a class="tab-item">
    <i class="icon ion-gear-a"></i>
  </a>
</div>
```

顶部图标+文字选项卡

在 tabs 类后添加 ttabs-icon-top 类可设置顶部图标+文字选项卡。

```
<div class="tabs tabs-icon-top">
  <a class="tab-item" href="#">
    <i class="icon ion-home"></i>
    主页
  </a>
  <a class="tab-item" href="#">
    <i class="icon ion-star"></i>
    收藏
  </a>
  <a class="tab-item" href="#">
    <i class="icon ion-gear-a"></i>
    设置
  </a>
</div>
```

左侧图标+文字选项卡

在 tabs 类后添加 ttabs-icon-left 类可设置左侧图标+文字选项卡。

```
<div class="tabs tabs-icon-left">
  <a class="tab-item">
    <i class="icon ion-home"></i>
    主页
  </a>
  <a class="tab-item">
    <i class="icon ion-star"></i>
    收藏
  </a>
  <a class="tab-item">
    <i class="icon ion-gear-a"></i>
    设置
  </a>
</div>
```

条纹样式选项卡

可以在带有 tabs 的样式名的元素上添加 tabs-striped 来实现像 Android 风格的 tabs。也可以添加 tabs-top 来实现选项卡在页面顶部。

条纹选项卡颜色可通过 tabs-background-{color} 和 tabs-color-{color} 来控制，{color} 值可以是：default, light, stable, positive, calm, balanced, energized, assertive, royal, 或 dark。

注意：如果要再选项卡上设置头部标题，需要使用 has-tabs-top 类。

```
<div class="tabs-striped tabs-top tabs-background-positive tabs-col
    <div class="tabs">
      <a class="tab-item active" href="#">
        <i class="icon ion-home"></i>
        Test
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-star"></i>
        Favorites
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-gear-a"></i>
        Settings
      </a>
    </div>
  </div>
  <div class="tabs-striped tabs-color-assertive">
    <div class="tabs">
      <a class="tab-item active" href="#">
        <i class="icon ion-home"></i>
        Test
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-star"></i>
        Favorites
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-gear-a"></i>
        Settings
      </a>
    </div>
  </div>
```

运行效果如下：


条纹样式选项卡



ionic 网格(Grid)

ionic 的网格(Grid)和其他大部分框架有所不同，它采用了弹性盒子模型(Flexible Box Model) 。而且在移动端，基本上的手机都支持。row 样式指定行，col 样式指定列。

同等大小网格

在带有 row 的样式的元素里如果包含了 col 的样式，col 就会设置为同等大小。

以下实例中 row 的样式包含了 5 个 col 样式，每个 col 的宽度为 20%。

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

指定列宽

你可以设定一行中各个列的大小不一样。默认情况下，列都会被划分为同等大小。但你也可以按百分比来设置列的宽度（一行为 12 个网格）。

```
<div class="row">
  <div class="col col-50">.col.col-50</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col col-75">.col.col-75</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col col-75">.col.col-75</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

注意：实例中，每个 col 样式会自动添加上边框和灰色背景。

下面列出了指定列宽的一些百分比的样式名:

.col-10	10%
.col-20	20%
.col-25	25%
.col-33	33.3333%
.col-50	50%
.col-67	66.6666%
.col-75	75%
.col-80	80%
.col-90	90%

有偏移量的网格

列可以设置左侧偏移量，实例如下：

```
<div class="row">
  <div class="col col-33 col-offset-33">.col</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col col-33">.col</div>
  <div class="col col-33 col-offset-33">.col</div>
</div>

<div class="row">
  <div class="col col-33 col-offset-67">.col</div>
</div>
```

下面是一些百分比的偏移量样式名：

| .col-offset-10 | 10% | | .col-offset-20 | 20% | | .col-offset-25 | 25% | | .col-offset-33 | 33.3333% | | .col-offset-50 | 50% | | .col-offset-67 | 66.6666% | | .col-offset-75 | 75% | | .col-offset-80 | 80% | | .col-offset-90 | 90% |

纵向对齐网格

弹性盒子模型可以很容易设置列纵向对齐。纵向对齐包含顶部，中间部分，底部，可以应用到每一行的列，或者指定的某列。

实例中，最后一列设置了最高的内容用于更好的演示纵向对齐网格。

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row">
  <div class="col col-top">.col</div>
  <div class="col col-center">.col</div>
  <div class="col col-bottom">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-top">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-center">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-bottom">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>
```

响应式网格

手持设备屏幕在切换时，例如横屏，竖屏等。就需要设置每行的网格可以实现根据不同宽度自适应大小。

不同设备响应式类的样式如下:

| 响应式类 | 描述 | | .responsive-sm | 小于手机横屏 | | .responsive-md | 小于平板竖屏 | | .responsive-lg | 小于平板横屏 |

```
<div class="row responsive-sm">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

ionic 颜色

ionic 提供了很多颜色的配置，当然你可以根据自己的需要自定义颜色。

```
<ul class="list color-list-demo">
    <li class="item dark">
      light
      <span class="color-demo light-bg light-border"></span>
    </li>
    <li class="item stable-dark">
      stable
      <span class="color-demo stable-bg stable-border"></span>
    </li>
    <li class="item positive">
      positive
      <span class="color-demo positive-bg positive-border"></span>
    </li>
    <li class="item calm">
      calm
      <span class="color-demo calm-bg calm-border"></span>
    </li>
    <li class="item balanced">
      balanced
      <span class="color-demo balanced-bg balanced-border"></span>
    </li>
    <li class="item energized">
      energized
      <span class="color-demo energized-bg energized-border"></spar
    </li>
    <li class="item assertive">
      assertive
      <span class="color-demo assertive-bg assertive-border"></spar
    </li>
    <li class="item royal">
      royal
      <span class="color-demo royal-bg royal-border"></span>
    </li>
    <li class="item dark">
      dark
      <span class="color-demo dark-bg dark-border"></span>
    </li>
</ul>
```

实例运行结果：

颜色列表
light
stable
positive
calm
balanced
energized
assertive
royal
dark

ionic icon(图标)

ionic 也默认提供了许多的图标，大概有500多个。用法也非常的简单：

```
<i  class="icon icon ion-star"></i>
```

图标样式CDN地址：http://www.runoob.com/static/ionic/css/ionicons.min.css。

图标列表如下：

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-



-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

ionic JavaScript

ionic 上拉菜单(ActionSheet)

上拉菜单(ActionSheet)通过往上弹出的框，来让用户选择选项。

非常危险的选项会以高亮的红色来让人第一时间识别。你可以通过点击取消按钮或者点击空白的地方来让它消失。

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >

    <ion-pane>
        <ion-content >
            <h2 ng-click="show()">Action Sheet</h2>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

在代码中触发上拉菜单，需要在你的 angular 控制器中使用 $ionicActionSheet 服务：

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$ionicActionSheet','$timeo
    $scope.show = function() {

        var hideSheet = $ionicActionSheet.show({
            buttons: [
              { text: '<b>Share</b> This' },
              { text: 'Move' }
            ],
            destructiveText: 'Delete',
            titleText: 'Modify your album',
            cancelText: 'Cancel',
            cancel: function() {
                 // add cancel code..
               },
            buttonClicked: function(index) {
              return true;
            }
        });

        $timeout(function() {
            hideSheet();
        }, 2000);

    };
}])
```

运行效果如下图：



ionic 背景层

我们经常需要在 UI 上，例如在弹出框、加载框、其他弹出层中显示或隐藏背景层。

在组件中可以使用$ionicBackdrop.retain()来显示背景层，使用$ionicBackdrop.release()隐藏背景层。

每次调用retain后，背景会一直显示，直到调用release消除背景层。

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >
    <ion-pane>
        <ion-content >
            <h2 ng-click="action()">$ionicBackdrop</h2>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

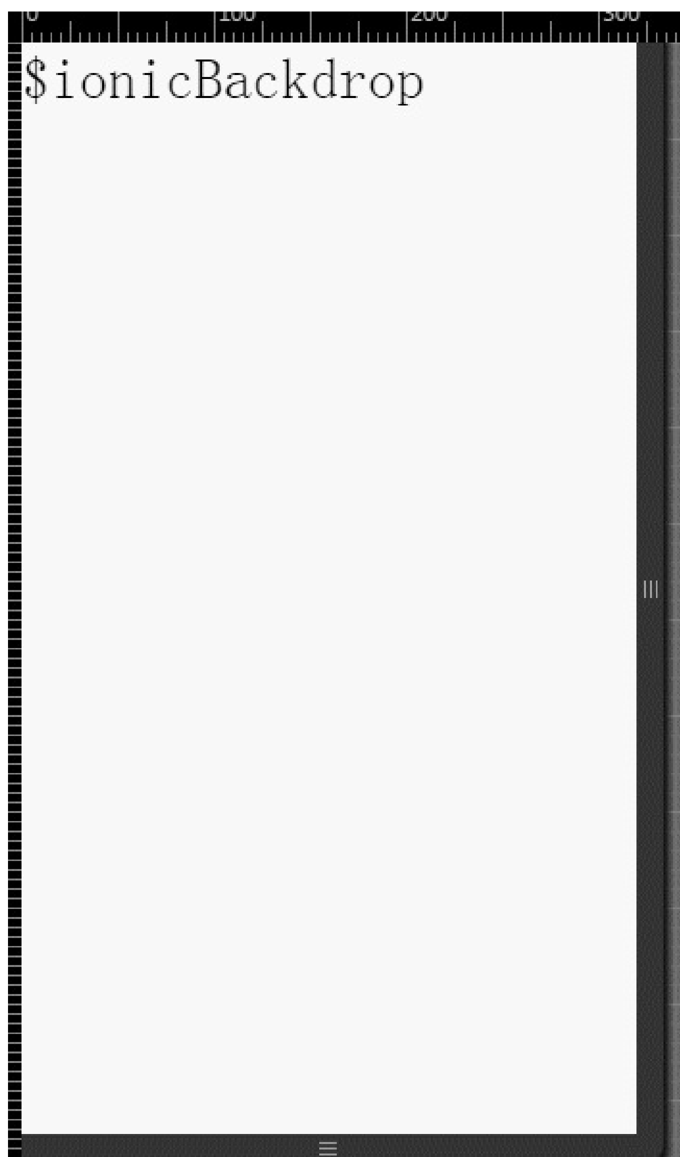
```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$timeout' ,'$ionicBackdrop

    $scope.action = function() {
       $ionicBackdrop.retain();
       $timeout(function() {    //默认让它1秒后消失
         $ionicBackdrop.release();
       }, 1000);
    };
}])
```

显示效果如下图所示：


$ionicBackdrop

ionic 下拉刷新

在加载新数据的时候，我们需要实现下拉刷新效果，代码如下：

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >
    <ion-pane>
        <ion-content >
            <ion-refresher pulling-text="下拉刷新" on-refresh="doRef
            <ion-list>
                <ion-item ng-repeat="item in items" ng-bind="item.r
            </ion-list>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$timeout' ,'$http',functio

    $scope.items=[
        {
            "name":"HTML5"
        },
        {
            "name":"JavaScript"
        },
        {
            "name":"Css3"
        }
    ];

    $scope.doRefresh = function() {
        $http.get('http://www.runoob.com/try/demo_source/item.json
            .success(function(newItems) {
                $scope.items = newItems;
            })
            .finally(function() {
                $scope.$broadcast('scroll.refreshComplete');
            });
    };
}])
```

item.json 文件数据：

```
[
    {
        "name":"菜鸟教程"
    },
    {
        "name":"www.runoob.com"
    }
]
```

效果如下所示：

HTML5
JavaScript
Css3

ionic 复选框

ionic 复选框（checkbox）与普通的 HTML 复选框没什么区别，以下实例演示了 ionic 复选框 ion-checkbox 的应用。

```
<ion-checkbox ng-model="isChecked">复选框标签</ion-checkbox>
```

实例

实例中，会根据复选框是否选中，修改 checked 值，true 为选中， false 为未选中。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">复选框</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <ion-checkbox ng-repeat="item in devList"
                  ng-model="item.checked"
                  ng-checked="item.checked">
      {{ item.text }}
    </ion-checkbox>

    <div class="item">
      <div ng-bind="devList | json"></div>
    </div>

    <div class="item item-divider">
      Notifications
    </div>

    <ion-checkbox ng-model="pushNotification.checked"
                  ng-change="pushNotificationChange()">
      Push Notifications
    </ion-checkbox>

    <div class="item">
      <div ng-bind="pushNotification | json"></div>
    </div>

    <ion-checkbox ng-model="emailNotification"
                  ng-true-value="'Subscribed'"
                  ng-false-value="'Unubscribed'">
      Newsletter
    </ion-checkbox>
    <div class="item">
      <div ng-bind="emailNotification | json"></div>
    </div>

  </div>

</ion-content>
```

JavaScript 代码

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
$ionicPlatform.ready(function() {
  // Hide the accessory bar by default (remove this to show the acc
  // for form inputs)
  if(window.cordova && window.cordova.plugins.Keyboard) {
    cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
  }
  if(window.StatusBar) {
    StatusBar.styleDefault();
  }
});
})

.controller( 'actionsheetCtl',['$scope',function($scope){

  $scope.devList = [
      { text: "HTML5", checked: true },
      { text: "CSS3", checked: false },
      { text: "JavaScript", checked: false }
    ];

    $scope.pushNotificationChange = function() {
      console.log('Push Notification Change', $scope.pushNotificat
    };

    $scope.pushNotification = { checked: true };
    $scope.emailNotification = 'Subscribed';

}])
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

复选框

 HTML5

 CSS3

 JavaScript

```
[
  {
    "text": "HTML5",
    "checked": true
  },
  {
    "text": "CSS3",
    "checked": false
  }
```

ionic 单选框操作

实例中，根据选中的不同选项，显示不同的值。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">当选按钮</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <div class="item item-divider">
      选取的值为: {{ data.clientSide }}
    </div>

    <ion-radio ng-repeat="item in clientSideList"
               ng-value="item.value"
               ng-model="data.clientSide">
      {{ item.text }}
    </ion-radio>

    <div class="item item-divider">
      Serverside, Selected Value: {{ data.serverSide }}
    </div>

    <ion-radio ng-repeat="item in serverSideList"
               ng-value="item.value"
               ng-model="data.serverSide"
               ng-change="serverSideChange(item)"
               name="server-side">
      {{ item.text }}
    </ion-radio>

  </div>

</ion-content>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MainCtrl', function($scope) {

  $scope.clientSideList = [
    { text: "Backbone", value: "bb" },
    { text: "Angular", value: "ng" },
    { text: "Ember", value: "em" },
    { text: "Knockout", value: "ko" }
  ];

  $scope.serverSideList = [
    { text: "Go", value: "go" },
    { text: "Python", value: "py" },
    { text: "Ruby", value: "rb" },
    { text: "Java", value: "jv" }
  ];

  $scope.data = {
    clientSide: 'ng'
  };

  $scope.serverSideChange = function(item) {
    console.log("Selected Serverside, text:", item.text, "value:",
  };

});
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

当选按钮

Backbone

Angular ✔

Ember

Knockout

Serverside, Selected Value:

Go

Python

ionic 切换开关操作

以下实例中，通过切换不同开关 checked 显示不同的值，true 为打开，false 为关闭。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">开关切换</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <div class="item item-divider">
      Settings
    </div>

    <ion-toggle ng-repeat="item in settingsList"
                ng-model="item.checked"
                ng-checked="item.checked">
      {{ item.text }}
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="settingsList | json"></div>
    </div>

    <div class="item item-divider">
      Notifications
    </div>

    <ion-toggle ng-model="pushNotification.checked"
                ng-change="pushNotificationChange()">
      Push Notifications
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="pushNotification | json"></div>
    </div>

    <ion-toggle toggle-class="toggle-assertive"
                ng-model="emailNotification"
                ng-true-value="Subscribed"
                ng-false-value="Unubscribed">
```

```
      Newsletter
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="emailNotification | json"></div>
    </div>

  </div>

</ion-content>
```

由于pre标签冲突，实例中的 pre 已替换为 div标签，具体可以在"尝试一下"中查看。

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MainCtrl', function($scope) {

  $scope.settingsList = [
    { text: "Wireless", checked: true },
    { text: "GPS", checked: false },
    { text: "Bluetooth", checked: false }
  ];

  $scope.pushNotificationChange = function() {
    console.log('Push Notification Change', $scope.pushNotificatior
  };

  $scope.pushNotification = { checked: true };
  $scope.emailNotification = 'Subscribed';

});
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

切换开关

Settings

Wireless

GPS

Bluetooth

```
[
  {
    "text": "Wireless",
    "checked": true
  },
  {
    "text": "GPS",
```

ionic 手势事件

`on-hold` ：长按的时间是500毫秒。

```
&lt;button
    on-hold="onHold()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-tap` ：这个是手势轻击事件，如果长按时间超过250毫秒，那就不是轻击了。。

```
&lt;button
    on-tap="onTap()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-double-tap` ：手双击屏幕事件

```
&lt;button
    on-double-tap="onDoubleTap()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-touch` ：这个和 on-tap 还是有区别的，这个是立即执行，而且是用户点击立马执行。不用等待 touchend/mouseup 。

```
    &lt;button on-touch="onTouch()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-release` ：当用户结束触摸事件时触发。

```
&lt;button
    on-release="onRelease()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag` ：这个有点类似于PC端的拖拽。当你一直点击某个物体，并且手开始移动，都会触发 on-drag。

```
&lt;button
    on-drag="onDrag()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-up` ：向上拖拽。

```
&lt;button
    on-drag-up="onDragUp()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-right` ：向右拖拽。

```
&lt;button
    on-drag-right="onDragRight()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-down` ：向下拖拽。

```
&lt;button
    on-drag-down="onDragDown()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-left` ：向左边拖拽。

```
&lt;button
    on-drag-left="onDragLeft()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe` ：指手指滑动效果，可以是任何方向上的。而且也和 on-drag 类似，都有四个方向上单独的事件。

```
&lt;button
    on-swipe="onSwipe()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-up` ：向上的手指滑动效果。

```
&lt;button
    on-swipe-up="onSwipeUp()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-right` ：向右的手指滑动效果。

```
&lt;button
    on-swipe-right="onSwipeRight()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-down` ：向下的手指滑动效果。

```
&lt;button
    on-swipe-down="onSwipeDown()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-left` ：向左的手指滑动效果。

```
&lt;button
    on-swipe-left="onSwipeLeft()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

$ionicGesture

一个angular服务展示ionicionic.EventController手势。

方法

```
on(eventType, callback, $element)
```

在一个元素上添加一个事件监听器。

`eventType` : `string`

监听的手势事件。

`callback` : `function(e)`

当手势事件发生时触发的事件。

`$element` : `element`

angular元素监听的事件。

`options` : `object`

对象。

```
off(eventType, callback, $element)
```

在一个元素上移除一个手势事件监听器。

`eventType` : `string`

移除监听的手势事件。

`callback` : `function(e)`

移除监听器。

`$element` : `element`

被监听事件的angular元素。

ionic 头部和底部

ion-header-bar

这个是固定在屏幕顶部的一个头部标题栏。如果给它加上'bar-subheader' 这个样式，它就是副标题。

用法

```
<ion-header-bar align-title="left" class="bar-positive">
  <div class="buttons">
    <button class="button" ng-click="doSomething()">Left Button</bu
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons">
    <button class="button">Right Button</button>
  </div>
</ion-header-bar>
<ion-content>
  Some content!
</ion-content>
```

API

`align-title_(optional)_` : `string`

这个是对齐 title 的。如果没有设置，它将会按照手机的默认排版(Ios的默认是居中，Android默认是居左)。它的值可以是 'left','center','right'。

`no-tap-scroll_(optional)_` : `boolean`

这个是设置 header-bar 是否跟随着内容的滚动而滚动，就是是否固定在顶部。它的值是布尔值（true/false）。

ion-footer-bar

知道了 ion-header-bar， 理解ion-footer-bar就轻松多啦！只是 ion-footer-bar 是在屏幕的底部。

用法

```
<ion-content>
  Some content!
</ion-content>
<ion-footer-bar align-title="left" class="bar-assertive">
  <div class="buttons">
    <button class="button">Left Button</button>
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons" ng-click="doSomething()">
    <button class="button">Right Button</button>
  </div>
</ion-footer-bar>
```

API

与 ion-header-bar 不同的是，ion-footer-bar 只有 align-title 这个 API。

`align-title(optional)` : `string`

这个是对齐 title 的。如果没有设置，它将会按照手机的默认排版(Ios的默认是居中，Android默认是居左)。它的值可以是 'left','center','right'。

ionic 列表操作

列表是一个应用广泛在几乎所有移动app中的界面元素。ionList 和 ionItem 这两个指令还支持多种多样的交互模式，比如移除其中的某一项，拖动重新排序，滑动编辑等等。

用法

```
<ion-list>
  <ion-item ng-repeat="item in items">
    Hello, {{item}}!
  </ion-item>
</ion-list>
```

高级用法: 缩略图，删除按钮，重新排序，滑动

```
<ion-list ng-controller="MyCtrl"
          show-delete="shouldShowDelete"
          show-reorder="shouldShowReorder"
          can-swipe="listCanSwipe">
  <ion-item ng-repeat="item in items"
            class="item-thumbnail-left">

    <img ng-src="{{item.img}}">
    <h2>{{item.title}}</h2>
    <p>{{item.description}}</p>
    <ion-option-button class="button-positive"
                       ng-click="share(item)">
      分享
    </ion-option-button>
    <ion-option-button class="button-info"
                       ng-click="edit(item)">
      编辑
    </ion-option-button>
    <ion-delete-button class="ion-minus-circled"
                       ng-click="items.splice($index, 1)">
    </ion-delete-button>
    <ion-reorder-button class="ion-navicon"
                        on-reorder="reorderItem(item, $fromIndex, $
    </ion-reorder-button>

  </ion-item>
</ion-list>
```

API

`delegate-handle(可选)` : `字符串`

该句柄定义带有 `$ionicListDelegate` 的列表。

`show-delete(可选)` : `布尔值`

列表项的删除按钮当前是显示还是隐藏。

`show-reorder(可选)` : `布尔值`

列表项的排序按钮当前是显示还是隐藏。

`can-swipe(可选)` : `布尔值`

列表项是否被允许滑动显示选项按钮。默认：true。

实例

HTML 代码：

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scal
    <title>Ionic List Directive</title>

    <link href="http://www.runoob.com/static/ionic/css/ionic.min.cs
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>

  <body ng-controller="MyCtrl">

    <ion-header-bar class="bar-positive">
      <div class="buttons">
        <button class="button button-icon icon ion-ios-minus-outlir
          ng-click="data.showDelete = !data.showDelete; data.showRe
      </div>
      <h1 class="title">Ionic Delete/Option Buttons</h1>
      <div class="buttons">
        <button class="button" ng-click="data.showDelete = false; c
            Reorder
        </button>
      </div>
    </ion-header-bar>

    <ion-content>

      <!-- The list directive is great, but be sure to also checkou
```

```
      <ion-list show-delete="data.showDelete" show-reorder="data.sh

        <ion-item ng-repeat="item in items"
                  item="item"
                  href="#/item/{{item.id}}" class="item-remove-anim
          Item {{ item.id }}
          <ion-delete-button class="ion-minus-circled"
                             ng-click="onItemDelete(item)">
          </ion-delete-button>
          <ion-option-button class="button-assertive"
                             ng-click="edit(item)">
            Edit
          </ion-option-button>
          <ion-option-button class="button-calm"
                             ng-click="share(item)">
            Share
          </ion-option-button>
          <ion-reorder-button class="ion-navicon" on-reorder="moveI
        </ion-item>

      </ion-list>

    </ion-content>

  </body>
</html>
```

CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MyCtrl', function($scope) {

  $scope.data = {
    showDelete: false
  };

  $scope.edit = function(item) {
    alert('Edit Item: ' + item.id);
```

```
  };
  $scope.share = function(item) {
    alert('Share Item: ' + item.id);
  };

  $scope.moveItem = function(item, fromIndex, toIndex) {
    $scope.items.splice(fromIndex, 1);
    $scope.items.splice(toIndex, 0, item);
  };

  $scope.onItemDelete = function(item) {
    $scope.items.splice($scope.items.indexOf(item), 1);
  };

  $scope.items = [
    { id: 0 },
    { id: 1 },
    { id: 2 },
    { id: 3 },
    { id: 4 },
    { id: 5 },
    { id: 6 },
    { id: 7 },
    { id: 8 },
    { id: 9 },
    { id: 10 },
    { id: 11 },
    { id: 12 },
    { id: 13 },
    { id: 14 },
    { id: 15 },
    { id: 16 },
    { id: 17 },
    { id: 18 },
    { id: 19 },
    { id: 20 },
    { id: 21 },
    { id: 22 },
    { id: 23 },
    { id: 24 },
    { id: 25 },
    { id: 26 },
    { id: 27 },
    { id: 28 },
    { id: 29 },
    { id: 30 },
    { id: 31 },
    { id: 32 },
    { id: 33 },
    { id: 34 },
    { id: 35 },
    { id: 36 },
    { id: 37 },
```

```
    { id: 38 },
    { id: 39 },
    { id: 40 },
    { id: 41 },
    { id: 42 },
    { id: 43 },
    { id: 44 },
    { id: 45 },
    { id: 46 },
    { id: 47 },
    { id: 48 },
    { id: 49 },
    { id: 50 }
  ];

});
```

ionic 加载动作

$ionicLoading 是 ionic 默认的一个加载交互效果。里面的内容也是可以在模板里面修改。

用法

```
angular.module('LoadingApp', ['ionic'])
.controller('LoadingCtrl', function($scope, $ionicLoading) {
  $scope.show = function() {
    $ionicLoading.show({
      template: 'Loading...'
    });
  };
  $scope.hide = function(){
    $ionicLoading.hide();
  };
});
```

方法

显示一个加载效果。

```
show(opts)
```

`opts` : `object`

loading指示器的选项。可用属性：

- `{string=}` `template` 指示器的html内容。
- `{string=}` `templateUrl` 一个加载html模板的url作为指示器的内容。
- `{boolean=}` `noBackdrop` 是否隐藏背景。默认情况下它会显示。
- `{number=}` `delay` 指示器延迟多少毫秒显示。默认为不延迟。
- `{number=}` `duration` 等待多少毫秒后自动隐藏指示器。默认情况下，指示器会一直显示，直到触发 `.hide()` 。

隐藏一个加载效果。

```
hide()
```

API

`delegate-handle(可选)` : `字符串`

该句柄定义带有 `$ionicListDelegate` 的列表。

`show-delete(可选)` : `布尔值`

列表项的删除按钮当前是显示还是隐藏。

`show-reorder(可选)` : `布尔值`

列表项的排序按钮当前是显示还是隐藏。

`can-swipe(可选)` : `布尔值`

列表项是否被允许滑动显示选项按钮。默认：true。

实例

HTML 代码：

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <title>Ionic Modal</title>

     <link href="http://www.runoob.com/static/ionic/css/ionic.min.c
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>
  <body ng-controller="AppCtrl">

      <ion-view title="Home">
        <ion-header-bar>
          <h1 class="title">The Stooges</h1>
        </ion-header-bar>
        <ion-content has-header="true">
          <ion-list>
            <ion-item ng-repeat="stooge in stooges" href="#">{{stoo
          </ion-list>
        </ion-content>
      </ion-view>

  </body>
</html>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])
.controller('AppCtrl', function($scope, $timeout, $ionicLoading) {

  // Setup the loader
  $ionicLoading.show({
    content: 'Loading',
    animation: 'fade-in',
    showBackdrop: true,
    maxWidth: 200,
    showDelay: 0
  });

  // Set a timeout to clear loader, however you would actually cal
  $timeout(function () {
    $ionicLoading.hide();
    $scope.stooges = [{name: 'Moe'}, {name: 'Larry'}, {name: 'Curly
  }, 2000);

});
```

$ionicLoadingConfig

设置加载的默认选项:

用法:

```
var app = angular.module('myApp', ['ionic'])
app.constant('$ionicLoadingConfig', {
  template: '默认加载模板……'
});
app.controller('AppCtrl', function($scope, $ionicLoading) {
  $scope.showLoading = function() {
    $ionicLoading.show(); //配置选项在 $ionicLoadingConfig 设置
  };
});
```

ionic 模型

$ionicModal

$ionicModal 可以遮住用户主界面的内容框。

你可以在你的 index 文件或者是其他文件内嵌入以下代码(里面的代码可以根据你自己的业务场景相应的改变)。

```
<script id="my-modal.html" type="text/ng-template">
  <ion-modal-view>
    <ion-header-bar>
      <h1 class="title">My Modal title</h1>
    </ion-header-bar>
    <ion-content>
      Hello!
    </ion-content>
  </ion-modal-view>
</script>
```

然后你就可以在你的 Controller 里面的注入 $ionicModal 。然后调用你刚刚写入的模板，进行初始化操作。就像下面的代码：

```
angular.module('testApp', ['ionic'])
.controller('MyController', function($scope, $ionicModal) {
  $ionicModal.fromTemplateUrl('my-modal.html', {
    scope: $scope,
    animation: 'slide-in-up'
  }).then(function(modal) {
    $scope.modal = modal;
  });
  $scope.openModal = function() {
    $scope.modal.show();
  };
  $scope.closeModal = function() {
    $scope.modal.hide();
  };
  //Cleanup the modal when we're done with it!
  $scope.$on('$destroy', function() {
    $scope.modal.remove();
  });
  // Execute action on hide modal
  $scope.$on('modal.hidden', function() {
    // Execute action
  });
  // Execute action on remove modal
  $scope.$on('modal.removed', function() {
    // Execute action
  });
});
```

方法

```
fromTemplate(templateString, options)
```

`templateString` : `字符串`

模板的字符串作为模型的内容。

`options` : `对象`

传递 ionicModal#initialize方法的选项。

返回: 对象, 一个ionicModal控制器的实例。

```
fromTemplateUrl(templateUrl, options)
```

`templateUrl` : `字符串`

载入模板的url。

`options` : `对象`

通过ionicModal#initialize方法传递对象。

返回： promise对象。Promises对象是CommonJS工作组提出的一种规范，目的是为异步编程提供统一接口。

ionicModal

由$ionicModal服务实例化。

提示：当你完成每个模块清除时，确保调用remove()方法，以避免内存泄漏。

注意：一个模块从它的初始范围广播出 'modal.shown' 和 'modal.hidden'，把自身作为一个参数来传递。

方法

```
initialize(可选)
```

创建一个新的模型控制器示例。

`options` : `对象`

一个选项对象具有一下属性：

- `{object=}` `范围` 子类的范围。默认：创建一个$rootScope子类。
- `{string=}` `动画` 带有显示或隐藏的动画。默认：'slide-in-up'
- `{boolean=}` `第一个输入框获取焦点` 当显示时，模型的第一个输入元素是否自动获取焦点。默认：false。
- `{boolean=}` backdropClickToClose` 点击背景时是否关闭模型。默认：true。

```
show()
```

显示模型实例

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
hide()
```

隐藏模型。

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
remove()
```

从 DOM 中移除模型实例并清理。

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
isShown()
```

- 返回：布尔值，用于判断模型是否显示。

实例

HTML 代码

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <title>菜鸟教程(runoob.com)</title>
    <link href="http://www.runoob.com/static/ionic/css/ionic.min.cs
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>
  <body ng-controller="AppCtrl">

    <ion-header-bar class="bar-positive">
      <h1 class="title">Contacts</h1>
      <div class="buttons">
        <button class="button button-icon ion-compose" ng-click="mo
        </button>
      </div>
    </ion-header-bar>
    <ion-content>
      <ion-list>
        <ion-item ng-repeat="contact in contacts">
          {{contact.name}}
        </ion-item>
      </ion-list>
    </ion-content>

    <script id="templates/modal.html" type="text/ng-template">
      <ion-modal-view>
        <ion-header-bar class="bar bar-header bar-positive">
          <h1 class="title">New Contact</h1>
          <button class="button button-clear button-primary" ng-cli
        </ion-header-bar>
        <ion-content class="padding">
```

```
            <div class="list">
              <label class="item item-input">
                <span class="input-label">First Name</span>
                <input ng-model="newUser.firstName" type="text">
              </label>
              <label class="item item-input">
                <span class="input-label">Last Name</span>
                <input ng-model="newUser.lastName" type="text">
              </label>
              <label class="item item-input">
                <span class="input-label">Email</span>
                <input ng-model="newUser.email" type="text">
              </label>
              <button class="button button-full button-positive" ng-c
            </div>
          </ion-content>
        </ion-modal-view>
      </script>

  </body>
</html>
```

CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('AppCtrl', function($scope, $ionicModal) {

  $scope.contacts = [
    { name: 'Gordon Freeman' },
    { name: 'Barney Calhoun' },
    { name: 'Lamarr the Headcrab' },
  ];

  $ionicModal.fromTemplateUrl('templates/modal.html', {
    scope: $scope
  }).then(function(modal) {
    $scope.modal = modal;
  });

  $scope.createContact = function(u) {
    $scope.contacts.push({ name: u.firstName + ' ' + u.lastName });
    $scope.modal.hide();
  };

});
```

ionic 导航

ion-nav-view

当用户在你的app中浏览时，ionic能够检测到浏览历史。通过检测浏览历史，实现向左或向右滑动时可以正确转换视图。

采用AngularUI路由器模块等应用程序接口可以分为不同的$state(状态)。Angular的核心为路由服务，URLs可以用来控制视图。

AngularUI路由提供了一个更强大的状态管理，即状态可以被命名，嵌套， 以及合并视图，允许一个以上模板呈现在同一个页面。

此外，每个状态无需绑定到一个URL，并且数据可以更灵活地推送到每个状态。

以下实例中，我们将创建一个应用程序中包含不同状态的导航视图。

我们的标记中选择ionNavView作为顶层指令。显示一个页眉栏我们用 ionNavBar 指令通过导航更新。

接下来，我们需要设置我们的将渲染的状态值。

```
var app = angular.module('myApp', ['ionic']);
app.config(function($stateProvider) {
  $stateProvider
  .state('index', {
    url: '/',
    templateUrl: 'home.html'
  })
  .state('music', {
    url: '/music',
    templateUrl: 'music.html'
  });
});
```

再打开应用，$stateProvider 会查询url, 看是否匹配 index 状态值, 再加载 index.html到<ion-nav-view>。

页面加载都是通过URLs配置的。在Angular中创建模板最一个简单的方式就是直接将他放到html模板文件中并且用

ionic 平台

$ionicPlatform

$ionicPlatform 用来检测当前的平台，以及诸如在PhoneGap/Cordova中覆盖Android后退按钮。

方法

```
onHardwareBackButton(callback)
```

有硬件的后退按钮的平台，可以用这种方法绑定到它。

`callback` : `function`

当该事件发生时，触发回调函数。

```
offHardwareBackButton(callback)
```

移除后退按钮的监听事件。

`callback` : `function`

最初绑定的监视器函数。

```
registerBackButtonAction(callback, priority, [actionId])
```

注册硬件后退按钮动作。当点击按钮时，只有一个动作会执行，因此该方法决定了注册的后退按钮动作具有最高的优先级。

例如，如果一个上拉菜单已经显示，后退按钮应该关闭上拉菜单，而不是返回一个页面视图或关闭一个打开的模型。

`callback` : `function`

当点击返回按钮时触发，如果该监视器具有最高的优先级。

`priority` : `number`

仅最高优先级的会执行。

`actionId(可选)` : `*`

该id指定这个动作。默认：一个随机且唯一的id。

返回值: 函数， 一个被触发的函数，将会注销 backButtonAction。

```
ready([callback])
```

设备准备就绪，则触发一个回调函数。

`callback(可选)` : `function=`

触发的函数。

返回: promise对象, 对象被构造 成功后得到解析。

ionic 浮动框

$ionicPopover

$ionicPopover 是一个可以浮在app内容上的一个视图框。

实例

HTML 代码

```
<p>
<button ng-click="openPopover($event)">打开浮动框</button>
</p>
<script id="my-popover.html" type="text/ng-template">
<ion-popover-view>
  <ion-header-bar>
    <h1 class="title">我的浮动框标题</h1>
  </ion-header-bar>
  <ion-content>
    Hello!
  </ion-content>
</ion-popover-view>
</script>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])
.controller( 'AppCtrl',['$scope','$ionicPopover','$timeout',functio

  $scope.popover = $ionicPopover.fromTemplateUrl('my-popover.html',
    scope: $scope
  });

  // .fromTemplateUrl() 方法
  $ionicPopover.fromTemplateUrl('my-popover.html', {
    scope: $scope
  }).then(function(popover) {
    $scope.popover = popover;
  });

  $scope.openPopover = function($event) {
    $scope.popover.show($event);
  };
  $scope.closePopover = function() {
    $scope.popover.hide();
  };
  // 清除浮动框
  $scope.$on('$destroy', function() {
    $scope.popover.remove();
  });
  // 在隐藏浮动框后执行
  $scope.$on('popover.hidden', function() {
    // 执行代码
  });
  // 移除浮动框后执行
  $scope.$on('popover.removed', function() {
    // 执行代码
  });

}])
```

ionic 对话框

$ionicPopup

ionic 对话框服务允许程序创建、显示弹出窗口。

$ionicPopup 提供了3个方法：alert(), prompt(),以及 confirm() 。

实例

HTML 代码

```
<body class="padding" ng-controller="PopupCtrl">
    <button class="button button-dark" ng-click="showPopup()">
      弹窗显示
    </button>
    <button class="button button-primary" ng-click="showConfirm()">
      确认对话框
    </button>
    <button class="button button-positive" ng-click="showAlert()">
      警告框
    </button>

    <script id="popup-template.html" type="text/ng-template">
      <input ng-model="data.wifi" type="text" placeholder="Passwor
    </script>
</body>
```

JavaScript 代码

```
angular.module('mySuperApp', ['ionic'])
.controller('PopupCtrl',function($scope, $ionicPopup, $timeout) {

 // Triggered on a button click, or some other target
 $scope.showPopup = function() {
   $scope.data = {}

   // 自定义弹窗
   var myPopup = $ionicPopup.show({
     template: '<input type="password" ng-model="data.wifi">',
     title: 'Enter Wi-Fi Password',
     subTitle: 'Please use normal things',
     scope: $scope,
```

```
      buttons: [
        { text: 'Cancel' },
        {
          text: '<b>Save</b>',
          type: 'button-positive',
          onTap: function(e) {
            if (!$scope.data.wifi) {
              // 不允许用户关闭，除非输入 wifi 密码
              e.preventDefault();
            } else {
              return $scope.data.wifi;
            }
          }
        },
      ]
    });
    myPopup.then(function(res) {
      console.log('Tapped!', res);
    });
    $timeout(function() {
       myPopup.close(); // 3秒后关闭弹窗
    }, 3000);
   };
    //  confirm 对话框
    $scope.showConfirm = function() {
      var confirmPopup = $ionicPopup.confirm({
        title: 'Consume Ice Cream',
        template: 'Are you sure you want to eat this ice cream?'
      });
      confirmPopup.then(function(res) {
        if(res) {
          console.log('You are sure');
        } else {
          console.log('You are not sure');
        }
      });
    };

    //  alert（警告） 对话框
    $scope.showAlert = function() {
      var alertPopup = $ionicPopup.alert({
        title: 'Don\'t eat that!',
        template: 'It might taste good'
      });
      alertPopup.then(function(res) {
        console.log('Thank you for not eating my delicious ice crear
      });
    };
 });
```

ionic 滚动条

ion-scroll

ion-scroll 用于创建一个可滚动的容器。

用法

```
<ion-scroll
    [delegate-handle=""]
    [direction=""]
    [paging=""]
    [on-refresh=""]
    [on-scroll=""]
    [scrollbar-x=""]
    [scrollbar-y=""]
    [zooming=""]
    [min-zoom=""]
    [max-zoom=""]>
    ...
</ion-scroll>
```

API

`delegate-handle(可选)` : `字符串`

该句柄利用 `$ionicScrollDelegate` 指定滚动视图。

`direction(可选)` : `字符串`

滚动的方向。 'x' 或 'y'。 默认 'y'。

`paging(可选)` : `布尔值`

分页是否滚动。

`on-refresh(可选)` : `表达式`

调用下拉刷新， 由 `ionRefresher` 触发。

`on-scroll(可选)` : `表达式`

当用户滚动时触发。

`scrollbar-x(可选)` : `布尔值`

是否显示水平滚动条。默认为false。

`scrollbar-y(可选)` : `布尔值`

是否显示垂直滚动条。默认为true。

`zooming(可选)` : `布尔值`

是否支持双指缩放。

`min-zoom(可选)` : `整数`

允许的最小缩放量（默认为0.5）

`max-zoom(可选)` : `整数`

允许的最大缩放量（默认为3）

实例

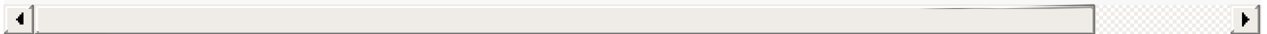
HTML 代码

```
<ion-scroll zooming="true" direction="xy" style="width: 500px; heig
    <div style="width: 5000px; height: 5000px; background: url('htt
</ion-scroll>
```



CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```



JavaScript 代码

```
angular.module('ionicApp', ['ionic']);
```

ion-infinite-scroll

当用户到达页脚或页脚附近时，ionInfiniteScroll指令允许你调用一个函数 。

当用户滚动的距离超出底部的内容时，就会触发你指定的on-infinite。

用法

```
<ion-content ng-controller="MyController">
  <ion-infinite-scroll
    on-infinite="loadMore()"
    distance="1%">
  </ion-infinite-scroll>
</ion-content>
```

```
function MyController($scope, $http) {
  $scope.items = [];
  $scope.loadMore = function() {
    $http.get('/more-items').success(function(items) {
      useItems(items);
      $scope.$broadcast('scroll.infiniteScrollComplete');
    });
  };

  $scope.$on('stateChangeSuccess', function() {
    $scope.loadMore();
  });
}
```

当没有更多数据加载时，就可以用一个简单的方法阻止无限滚动，那就是angular的ng-if 指令:

```
<ion-infinite-scroll
  ng-if="moreDataCanBeLoaded()"
  icon="ion-loading-c"
  on-infinite="loadMoreData()">
</ion-infinite-scroll>
```

API

`on-infinite` ： `表达式`

当滚动到底部时触发的时间。

`distance(可选)` ： `字符串`

从底部滚动到触发on-infinite表达式的距离。默认: 1%。

`icon(可选)` ： `字符串`

当加载时显示的图标。默认: 'ion-loading-d'。

$ionicScrollDelegate

授权控制滚动视图（通过ion-content 和 ion-scroll指令创建）。

该方法直接被$ionicScrollDelegate服务触发，来控制所有滚动视图。用$getByHandle方法控制特定的滚动视图。

用法

```
<body ng-controller="MainCtrl">
  <ion-content>
    <button ng-click="scrollTop()">滚动到顶部!</button>
  </ion-content>
</body>
```

```
function MainCtrl($scope, $ionicScrollDelegate) {
  $scope.scrollTop = function() {
    $ionicScrollDelegate.scrollTop();
  };
}
```

方法

```
resize()
```

告诉滚动视图重新计算它的容器大小。

```
scrollTop([shouldAnimate])
```

`shouldAnimate(可选)` ： `布尔值`

是否应用滚动动画。

```
scrollBottom([shouldAnimate])
```

`shouldAnimate(可选)` ： `布尔值`

是否应用滚动动画。

ionic 侧栏菜单

一个容器元素包含侧边菜单和主要内容。通过把主要内容区域从一边拖动到另一边，来让左侧或右侧的侧栏菜单进行切换。

效果图如下所示：



用法

要使用侧栏菜单，添加一个父元素<ion-side-menus>，一个中间内容 <ion-side-menu-content>，和一个或更多 <ion-side-menu> 指令。

```
<ion-side-menus>
  <!-- 中间内容 -->
  <ion-side-menu-content ng-controller="ContentController">
  </ion-side-menu-content>

  <!-- 左侧菜单 -->
  <ion-side-menu side="left">
  </ion-side-menu>

  <!-- 右侧菜单 -->
  <ion-side-menu side="right">
  </ion-side-menu>
</ion-side-menus>
```

```
function ContentController($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeft = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

API

`enable-menu-with-back-views(可选)` ： `布尔值`

在返回按钮显示时，确认是否启用侧边栏菜单。

`delegate-handle` ：字符串 该句柄用于标识带有$ionicScrollDelegate的滚动视图。

ion-side-menu-content

一个可见主体内容的容器，同级的一个或多个ionSideMenu 指令。

用法

```
<ion-side-menu-content
  drag-content="true">
</ion-side-menu-content>
```

API

`drag-content(可选)` ： `布尔值`

内容是否可被拖动。默认为true。

ion-side-menu

一个侧栏菜单的容器，同级的一个ion-side-menu-content 指令。

用法

```
<ion-side-menu
  side="left"
  width="myWidthValue + 20"
  is-enabled="shouldLeftSideMenuBeEnabled()">
</ion-side-menu>
```

API

`side` ： `字符串`

侧栏菜单当前在哪一边。可选的值有: 'left' 或 'right'。

`is-enabled(可选)` ： `布尔值`

该侧栏菜单是否可用。

`width(可选)` ： `数值`

侧栏菜单应该有多少像素的宽度。默认为275。

menu-toggle

在一个指定的侧栏中切换菜单。

用法

下面是一个在导航栏内链接的例子。点击此链接会自动打开指定的侧栏菜单。

```
<ion-view>
  <ion-nav-buttons side="left">
   <button menu-toggle="left" class="button button-icon icon ion-na
  </ion-nav-buttons>
 ...
</ion-view>
```

menu-close

关闭当前打开的侧栏菜单。

用法

下面是一个在导航栏内链接的例子。点击此链接会自动打开指定的侧栏菜单。

```
<a menu-close href="#/home" class="item">首页</a>
```

$ionicSideMenuDelegate

该方法直接触发$ionicSideMenuDelegate服务，来控制所有侧栏菜单。用$getByHandle方法控制特定情况下的ionSideMenus。

用法

```
<body ng-controller="MainCtrl">
  <ion-side-menus>
    <ion-side-menu-content>
      内容!
      <button ng-click="toggleLeftSideMenu()">
        切换左侧侧栏菜单
      </button>
    </ion-side-menu-content>
    <ion-side-menu side="left">
      左侧菜单!
    <ion-side-menu>
  </ion-side-menus>
</body>
```

```
function MainCtrl($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeftSideMenu = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

方法

```
toggleLeft([isOpen])
```

切换左侧侧栏菜单（如果存在）。

`isOpen(可选)` : `布尔值`

是否打开或关闭菜单。默认：切换菜单。

```
toggleRight([isOpen])
```

切换右侧侧栏菜单（如果存在）。

`isOpen(可选)` ： `布尔值`

是否打开或关闭菜单。默认：切换菜单。

```
getOpenRatio()
```

获取打开菜单内容超出菜单宽度的比例。比如，一个宽度为100px的菜单被宽度为50px以50%的比例打开，将会返回一个比例值为0.5。

返回值: 浮点 0 表示没被打开，如果左侧菜单处于已打开或正在打开为0 到 1，如果右侧菜单处于已打开或正在打开为0 到-1。

```
isOpen()
```

返回值: 布尔值，判断左侧或右侧菜单是否已经打开。

```
isOpenLeft()
```

返回值: 布尔值左侧菜单是否已经打开。

```
isOpenRight()
```

返回值: 布尔值右侧菜单是否已经打开。

```
canDragContent([canDrag])
```

`canDrag(可选)` ： `布尔值`

设置是否可以拖动内容打开侧栏菜单。

返回值: 布尔值，是否可以拖动内容打开侧栏菜单。

```
$getByHandle(handle)
```

`handle` ： `字符串`

例如:

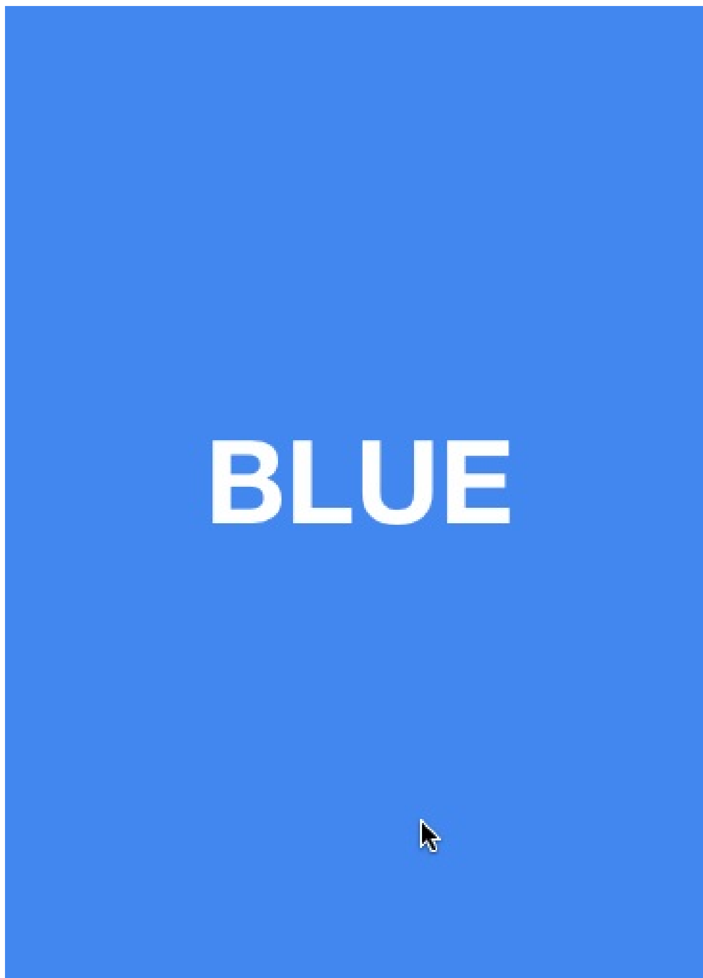
```
$ionicSideMenuDelegate.$getByHandle('my-handle').toggleLeft();
```

ionic 滑动框

ion-slide-box

滑动框是一个包含多页容器的组件，每页滑动或拖动切换：

效果图如下：



用法

```
<ion-slide-box on-slide-changed="slideHasChanged($index)">
  <ion-slide>
    <div class="box blue"><h1>BLUE</h1></div>
  </ion-slide>
  <ion-slide>
    <div class="box yellow"><h1>YELLOW</h1></div>
  </ion-slide>
  <ion-slide>
    <div class="box pink"><h1>PINK</h1></div>
  </ion-slide>
</ion-slide-box>
```

API

`delegate-handle(可选)` : `字符串`

该句柄用 `$ionicSlideBoxDelegate` 来标识这个滑动框。

`does-continue(可选)` : `布尔值`

滑动框是否自动滑动。

`slide-interval(可选)` : `数字`

等待多少毫秒开始滑动（如果继续则为true）。默认为4000。

`show-pager(可选)` : `布尔值`

滑动框的页面是否显示。

`pager-click(可选)` : `表达式`

当点击页面时，触发该表达式（如果shou-pager为true）。传递一个'索引'变量。

`on-slide-changed(可选)` : `表达式`

当滑动时，触发该表达式。传递一个'索引'变量。

`active-slide(可选)` : `表达式`

将模型绑定到当前滑动框。

实例

HTML 代码

```
<ion-slide-box active-slide="myActiveSlide">
    <ion-slide>
      <div class="box blue"><h1>BLUE</h1></div>
    </ion-slide>
    <ion-slide>
      <div class="box yellow"><h1>YELLOW</h1></div>
    </ion-slide>
    <ion-slide>
      <div class="box pink"><h1>PINK</h1></div>
    </ion-slide>
</ion-slide-box>
```

CSS 代码

```
.slider {
  height: 100%;
}
.slider-slide {
  padding-top: 80px;
  color: #000;
  background-color: #fff;
  text-align: center;

  font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Hel
  font-weight: 300;
}

.blue {
  background-color: blue;
}

.yellow {
  background-color: yellow;
}

.pink {
  background-color: pink;
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('SlideController', function($scope) {

  $scope.myActiveSlide = 1;

})
```

ionic 加载动画

ion-spinner

ionSpinner 提供了许多种旋转加载的动画图标。当你的界面加载时，你就可以呈现给用户相应的加载图标。

该图标采用的是SVG。

用法

```
<ion-spinner icon="spiral"></ion-spinner>    //默认用法
```

像大部分其他的ionic组件一样，spinner也可以使用ionic的标准颜色命名规则，就像下面这样：

```
<ion-spinner class="spinner-energized"></ion-spinner>
```

实例

HTML 代码

```
<ion-content scroll="false" class="has-header">
  <p>
    <ion-spinner icon="android"></ion-spinner>
    <ion-spinner icon="ios"></ion-spinner>
    <ion-spinner icon="ios-small"></ion-spinner>
    <ion-spinner icon="bubbles" class="spinner-balanced"></ion-spi
    <ion-spinner icon="circles" class="spinner-energized"></ion-sp
  </p>

  <p>
    <ion-spinner icon="crescent" class="spinner-royal"></ion-spinne

    <ion-spinner icon="dots" class="spinner-dark"></ion-spinner>
    <ion-spinner icon="lines" class="spinner-calm"></ion-spinner>
    <ion-spinner icon="ripple" class="spinner-assertive"></ion-spi
    <ion-spinner icon="spiral"></ion-spinner>
  </p>

</ion-content>
```

CSS 代码

```
body {
  cursor: url('http://www.runob.com/try/demo_source/finger.png'), a
}
p {
  text-align: center;
  margin-bottom: 40px !important;
}
.spinner svg {
  width: 19% !important;
  height: 85px !important;
}
```

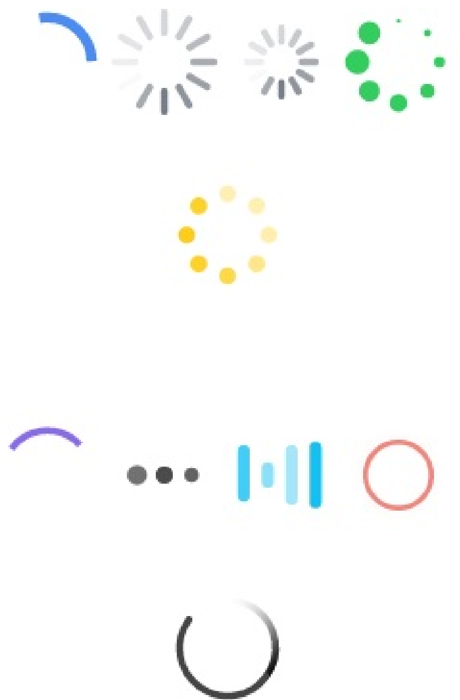
JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MyCtrl', function($scope) {

});
```

效果如下所示：



ionic 选项卡栏操作

ion-tabs

ion-tabs 是有一组页面选项卡组成的选项卡栏。可以通过点击选项来切换页面。

对于 iOS，它会出现在屏幕的底部，Android会出现在屏幕的顶部(导航栏下面)。

用法

```
<ion-tabs class="tabs-positive tabs-icon-only">

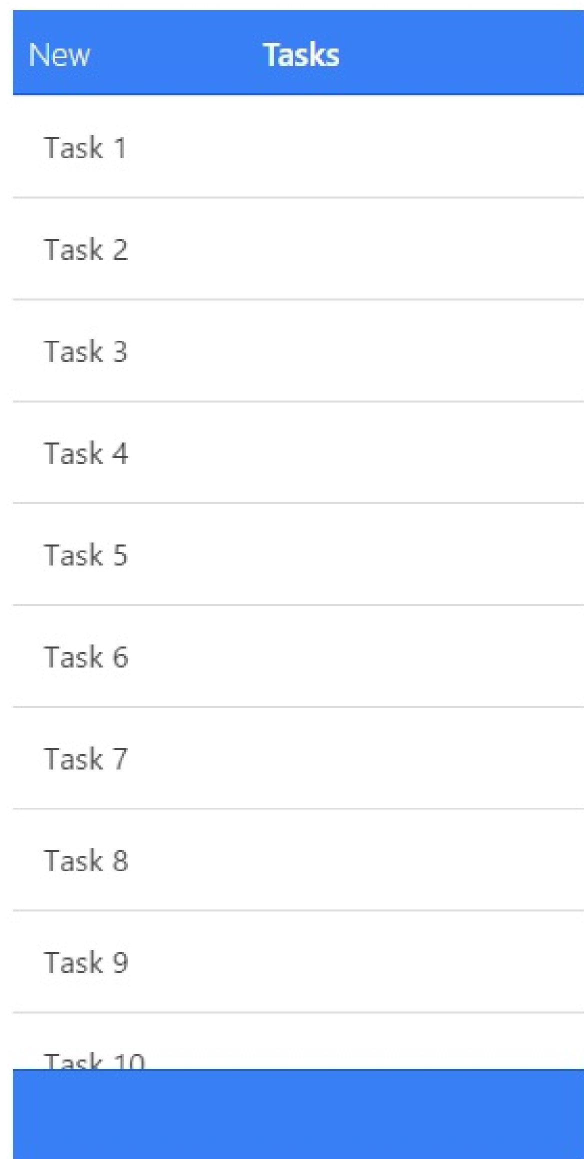
  <ion-tab title="首页" icon-on="ion-ios7-filing" icon-off="ion-ios
    <!-- 标签 1 内容 -->
  </ion-tab>

  <ion-tab title="关于" icon-on="ion-ios7-clock" icon-off="ion-ios7
    <!-- 标签 2 内容 -->
  </ion-tab>

  <ion-tab title="设置" icon-on="ion-ios7-gear" icon-off="ion-ios7-
    <!-- 标签 3 内容 -->
  </ion-tab>

</ion-tabs>
```

效果如下所示：



API

`delegate-handle(可选)` : `字符串`

该句柄用 `$ionicTabsDelegate` 来标识这些选项卡。

ion-tab

隶属于ionTabs

包含一个选项卡内容。该内容仅存在于被选中的给定选项卡中。

每个ionTab都有自己的浏览历史。

用法

```
<ion-tab
  title="Tab!"
  icon="my-icon"
  href="#/tab/tab-link"
  on-select="onTabSelected()"
  on-deselect="onTabDeselected()">
</ion-tab>
```

API

`title` ： `字符串`

选项卡的标题。

`href(可选)` ： `字符串`

但触碰的时候，该选项卡将会跳转的的链接。

`icon(可选)` ： `字符串`

选项卡的图标。如果给定值，它将成为ion-on和ion-off的默认值。

`icon-on(可选)` ： `字符串`

被选中标签的图标。

`icon-off(可选)` ： `字符串`

没被选中标签的图标。

`badge(可选)` ： `表达式`

选项卡上的徽章（通常是一个数字）。

`badge-style(可选)` ： `表达式`

选项卡上微章的样式（例，tabs-positive ）。

`on-select(可选)` ： `表达式`

选项卡被选中时触发。

`on-deselect(可选)` ： `表达式`

选项卡取消选中时触发。

`ng-click(可选)` ： `表达式`

通常，点击时选项卡会被选中。如果设置了 ng-Click，它将不会被选中。 你可以用$ionicTabsDelegate.select()来指定切换标签。

$ionicTabsDelegate

授权控制ionTabs指令。

该方法直接调用$ionicTabsDelegate服务，控制所有ionTabs指令。用$getByHandle方法控制具体的ionTabs实例。

用法

```
<body ng-controller="MyCtrl">
  <ion-tabs>

    <ion-tab title="Tab 1">
      你好，标签1！
      <button ng-click="selectTabWithIndex(1)">选择标签2</button>
    </ion-tab>
    <ion-tab title="Tab 2">你好标签2！</ion-tab>

  </ion-tabs>
</body>
```

```
function MyCtrl($scope, $ionicTabsDelegate) {
  $scope.selectTabWithIndex = function(index) {
    $ionicTabsDelegate.select(index);
  }
}
```

方法

```
select(index, [shouldChangeHistory])
```

选择标签来匹配给定的索引。

`index` ： `数值`

选择标签的索引。

`shouldChangeHistory(可选)` ： `布尔值`

此选项是否应该加载这个标签的浏览历史（如果存在），并使用，或仅加载默认页面。默认为false。提示：如果一个 `ion-nav-view` 在选项卡里，你可能需要设置它为true。

```
selectedIndex()
```

返回值: 数值, 被选中标签的索引，如 -1。

```
$getByHandle(handle)
```

`handle` ： `字符串`

例如:

```
$ionicTabsDelegate.$getByHandle('my-handle').select(0);
```
